Chapter 4

Geometric transformations

- geometric transformations deal with "moving pixels around" $\boxed{\rightarrow \texttt{Fig. 4.1}}$

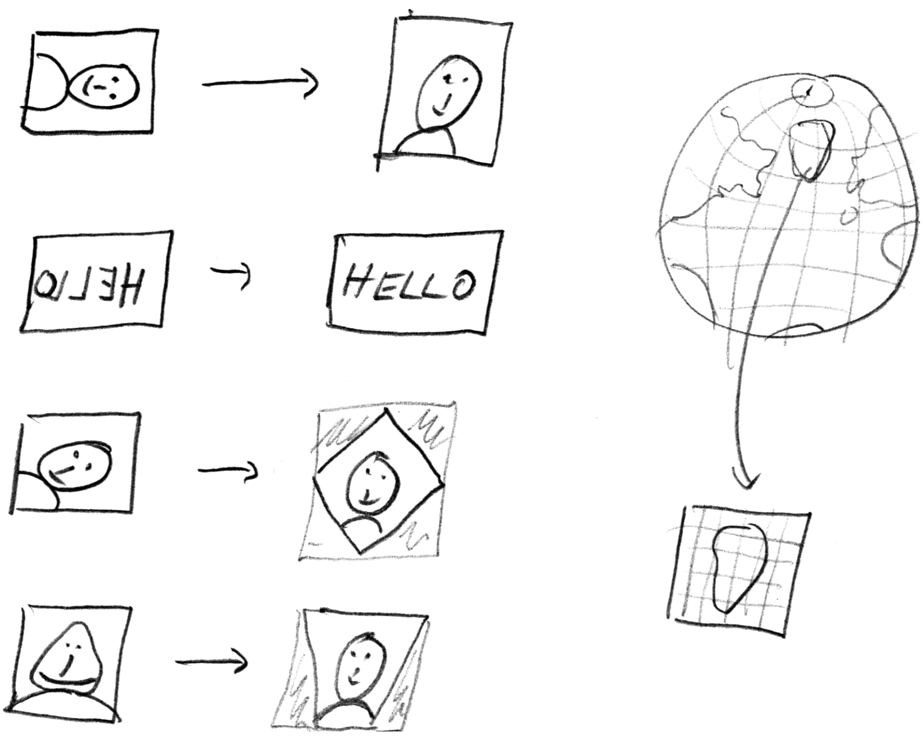

Fig. 4.1: *Some examples for geometric transformations: Rotations, mirroring, distortion, map projection.*

  - shift
  - rotation
  - scaling
  - other distortions

- Note: pixels of a digital image must be in a regular (Cartesian, equidistant) grid – i.e., pixel coordinates must be integer

- BUT: transformation might cause pixels to end up in a different grid – non-integer pixel coordinates →Fig. 4.2

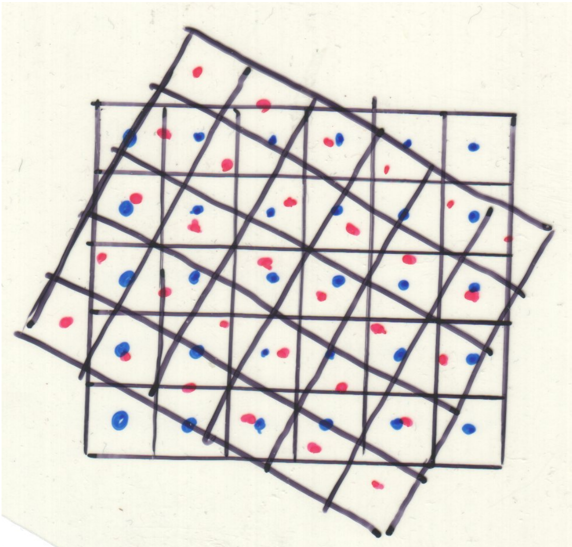

Fig. 4.2: *Pixels of rotated image (red) are between the positions of the original pixels (blue) – in other words, they have non-integer pixel coordinates.*

⇒ interpolation needed to get new pixels in a new regular grid with integer coordinates

⇒ Geometric transformation of digital images = two separate operations:

1. spatial transformation: rotation, shift, scaling etc.
2. grey-level interpolation:

$f(x,y)$: input image (original image)
$g(x,y)$: output image (resulting image after transformation)
x,y: integers (pixel coordinates)

$$g(x,y) = f(x',y') = f(a(x,y),b(x,y)) \tag{4.1}$$

$$\text{i.e., } x' = a(x,y) \tag{4.2}$$
$$y' = b(x,y) \tag{4.3}$$

4.1 Grey level interpolation

"Grey level" = pixel value
There are two ways to solve the interpolation problem:

1. forward-mapping approach (pixel carry-over):

- if a pixel from input image f maps to non-integer coordinates (between four integer pixel positions):

  divide its grey-level among these four pixels $\boxed{\rightarrow\texttt{Fig. 4.3}}$

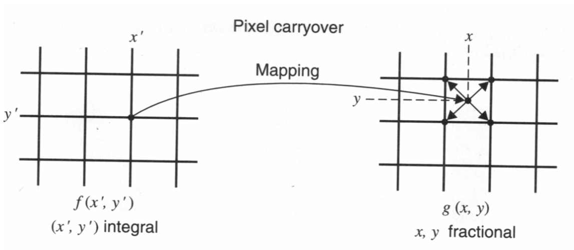

Fig. 4.3: *Pixel carry-over (from Castleman, 1996, Fig. 8.1)*

- Problem: Some areas in output image might not be reached

2. backward-mapping approach (pixel filling):

   - "allowed" output pixel positions in g are mapped back to original image f, one by one
   - if a pixel position falls between four input pixels in f, the pixel value is interpolated $\boxed{\rightarrow\texttt{Fig. 4.4}}$

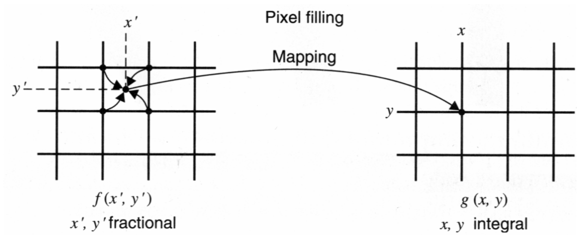

Fig. 4.4: *Pixel filling (from Castleman, 1996, Fig. 8.1)*

- pixel filling is the preferred method
- various schemes for the 2-dimensional interpolation:
  - nearest neighbor
  - bilinear
  - others…

Nearest neighbor interpolation

- the grey level of the nearest pixel is taken
- no computations necessary (fast!)

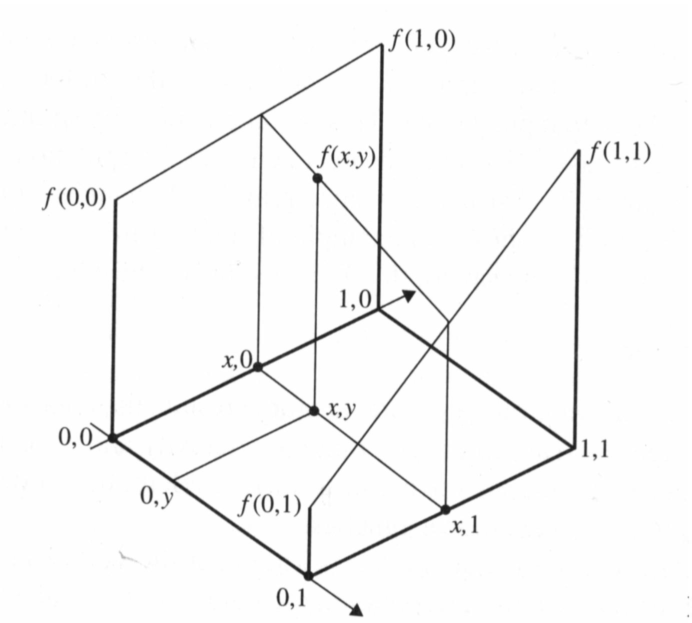

Fig. 4.5: *Bilinear interpolation (from Castleman, 1996, Fig. 8.2)*

Bilinear interpolation

→`Fig. 4.5`

- square with corners at $(0,0)$, $(1,0)$, $(0,1)$, and $(1,1)$
- values at the corners known, $f(0,0)$, $f(1,0)$, $f(0,1)$, $f(1,1)$
- wanted: value $f(x,y)$ at a given position (x,y) inside the square, i.e. x and y fixed and $0 \leq x,y \leq 1$
- first: linear interpolation first along two sides of the square

$$f(x,0) = f(0,0) + x[f(1,0) - f(0,0)] \tag{4.4}$$
$$f(x,1) = f(0,1) + x[f(1,1) - f(0,1)] \tag{4.5}$$

- then: linear interpolation between $f(x,0)$ and $f(x,1)$ to get to the point (x,y) inside the square:

$$f(x,y) = f(x,0) + y[f(x,1) - f(x,0)] \tag{4.6}$$

- Substitute Eq. s 4.4 and 4.5 into Eq. (4.6):

$$\begin{aligned} f(x,y) = \; & [f(1,0) - f(0,0)]x \\ & +[f(0,1 - f(0,0)]y \\ & +[f(1,1) + f(0,0) - f(0,1) - f(1,0)]xy \\ & +f(0,0) \end{aligned} \tag{4.7}$$

Note: this is the equation for a hyperbolic paraboloid ($f(x,y) = ax + by + cxy + d$)

Higher-order interpolation

Necessary, when bilinear interpolation causes too much smoothing

- cubic splines
- sinc function

(uses more than the surrounding 4 pixels)

4.2 Spatial Transformation

- Remember: $g(x,y) = f(x',y') = f(a(x,y),b(x,y))$
- output image value at position (x,y) = input image value at position (x',y')

4.2.1 Linear transformation, RST (rotation, scaling, translation)

- $a(x,y)$ and $b(x,y)$ are linear in x and y
- Formulation with “homogeneous coordinates”:

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.8}$$

Identity

$$x' = a(x,y) = x \; , \; y' = b(x,y) = y$$

i.e.

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.9}$$

Shift

$$x' = a(x,y) = x + x_0 \; , \; y' = b(x,y) = y + y_0$$

i.e.

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.10}$$

Scaling (expanding or shrinking)

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} 1/c & 0 & 0 \\ 0 & 1/d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.11}$$

- Image scaled in x-direction by factor c
- Image scaled in y-direction by factor d

Rotation

Rotation about origin $(0,0)$ by angle α:

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.12}$$

Combinations

- Compound transformations: matrix multiplication, sequence: right to left (for backward-mapping approach)
- e.g., rotation about (x_0, y_0): shift – rotate – shift back

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.13}$$

4.2.2 General Transformations, Control points

- till now: Rotation, scaling, translation (shift) (=RST), linear transformations: $a(x,y)$ and $b(x,y)$ are linear in x and y.
- more general transformations: polynomial transformations: $a(x,y)$ and $b(x,y)$ are polynomials of higher order than 1
- Often exact transformation is derived from control points
- control point = a point in the input image and its corresponding point in output image $\boxed{\rightarrow \text{Fig. 4.6}}$

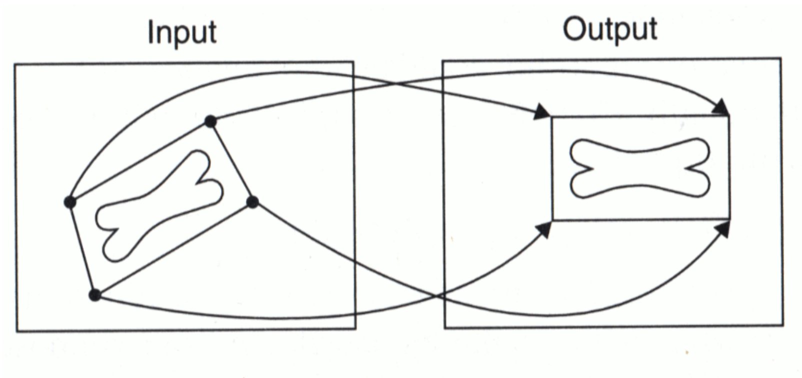

Fig. 4.6: *Control points (from Castleman, 1996, Fig. 8.5)*

- Two different setups for determining a general transformation from control points:
  1. Control grid interpolation: there is a regular array (grid) of control points
  2. Polynomial warping: control points are distributed irregularly

Control grid interpolation

- Control grid, e.g. from a test target
- distorted quadrilaterals map to regular rectangles $\boxed{\to\texttt{Fig. 4.7}}$

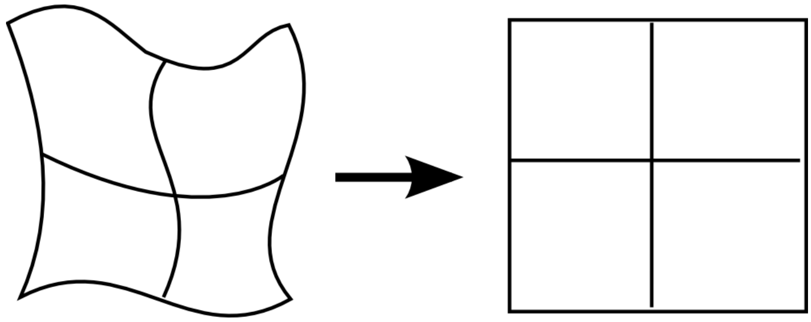

Fig. 4.7: *Quadrilaterals map to rectangles*

- Corners of quadrilaterals map to corners of rectangles
- Mapping of points inside quadrilaterals determined from interpolation, using the 4 corners (usually, bilinear interpolation), see Castleman (1996, chap. 8.3.5)
- Example application: Rectifying (or geometrically calibrating) an image taken with a fish-eye lens: $\boxed{\to\texttt{Fig. 4.8}}$

Polynomial warping

- assume $a(x,y)$ and $b(x,y)$ as polynomials of order N with unknown coefficients:

$$x' = a(x,y) = \sum_{i=0}^{N}\sum_{j=0}^{N-i} a_{ij}x^i y^j \tag{4.14}$$

$$y' = b(x,y) = \sum_{i=0}^{N}\sum_{j=0}^{N-i} b_{ij}x^i y^j \tag{4.15}$$

- Determining the coefficients requires at least as many control points as the polynomials have coefficients
- linear transformations mentioned above (RST) are contained in this general form for $N = 1$
- $N = 2$ sufficient for satellite image of few hundred kilometers size and small terrain relief:

$$x' = a(x,y) = a_{00} + a_{10}x + a_{01}y + a_{11}xy + a_{20}x^2 + a_{02}y^2 \tag{4.16}$$

$$y' = b(x,y) = b_{00} + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2 \tag{4.17}$$

The meaning of some of the coefficients $\boxed{\to\texttt{Fig. 4.9}}$

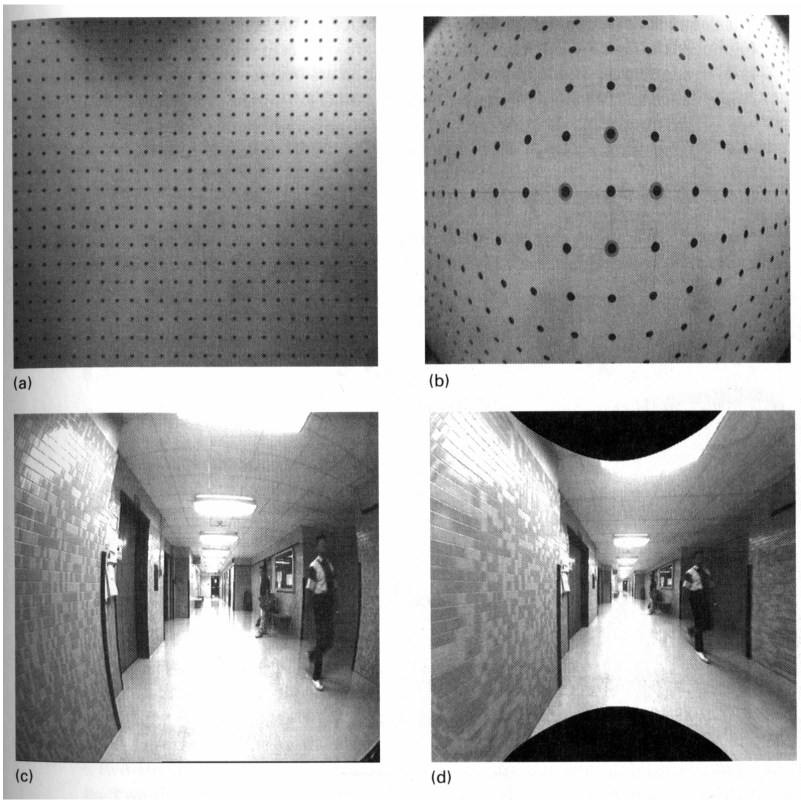

Fig. 4.8: Geometric rectification of an image taken with a fish-eye lens: (a) test target, (b) fish-eye image of test target, (c) fish-eye image (d) rectified image (Fig 8.9 from Castleman, 1996)

a_{00}, b_{00}: Shift vector

a_{10}, b_{01}: Linear scaling in x, y direction

a_{01}, b_{10}: Shear in x, y direction[1]

a_{11}, b_{11}: y-dependent scale in x, x-dependent scale in y

a_{20}, b_{02}: non-linear (quadratic) scale in x, y

- control points are often not exact ⇒no solution!?

⇒ use (many) more control points than needed to solve the equations, then do least square fit, i.e. find those coefficients that match best

- the control points should be distributed over the whole image

[1] A rotation can be described as a combination of shear and linear scaling first in one, then the other coordinate: Any Rotation by angle $\theta \neq \pm 90°$ can be decomposed in the following way:

$$\begin{bmatrix} a_{10} & a_{01} \\ b_{10} & b_{01} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1/\cos\theta & \sin\theta/\cos\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{4.18}$$

The first (the rightmost one) is a 1D scale and shear in y, the second (the left one) is a 1D scale and shear in x.

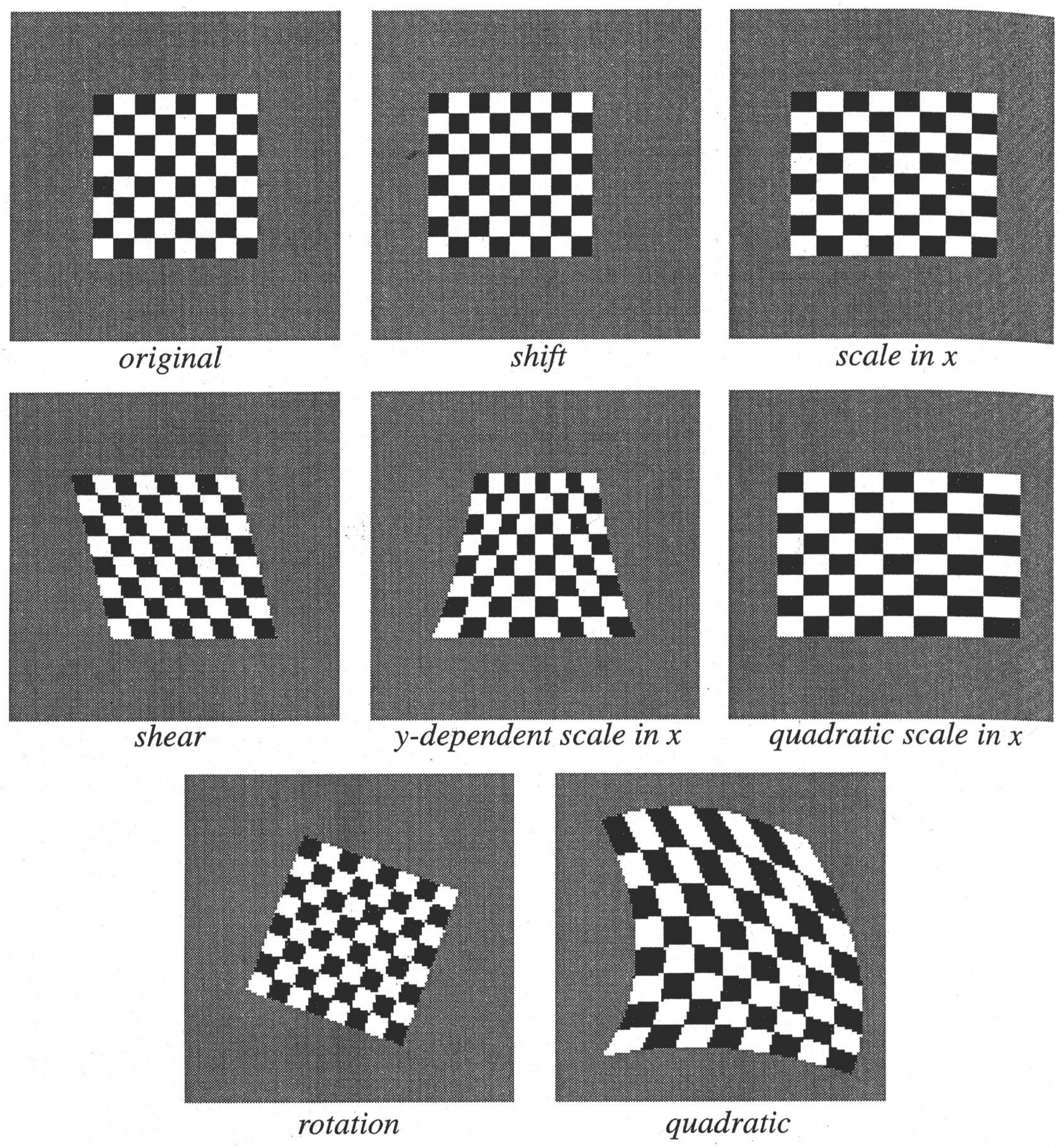

Fig. 4.9: *Some polynomial geometric warps (Fig 7-30 in Schowengerdt, 1997)*

4.2.3 Applications

Geometric calibration/Image Rectification: remove camera-induced distortion (Fig. 4.8), i.e., convert non-rectangular pixel coordinates to rectangular coordinates

Image registration: Geometrically match two images or an image and a map; stationary objects should have same position in both images (or in image and map) $\boxed{\rightarrow\texttt{Fig. 4.10}}$

Map projections

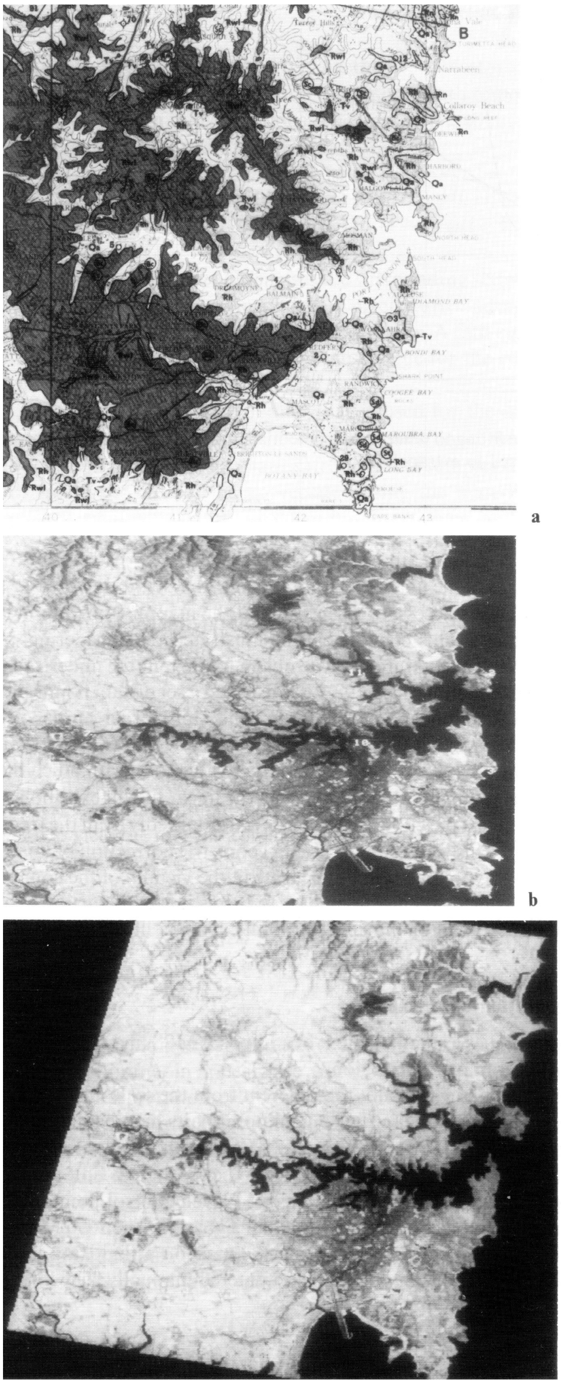

Fig. 4.10: *Image registration. (a) Map; (b) Landsat MSS image to be registered; (c) Landsat image registered to map using 2nd order polynomials (Fig. 2.16 from Richards, 1986)*

Bibliography

K. R. Castleman. *Digital Image Processing*. Prentice Hall, Englewood Cliffs, 1996.

J. A. Richards. *Remote Sensing Digital Image Analysis*. Springer, 1986. ISBN 0-387-16007-8.

R. A. Schowengerdt. *Remote Sensing, Models and Methods for Image Processing*. Academic Press, 1997.